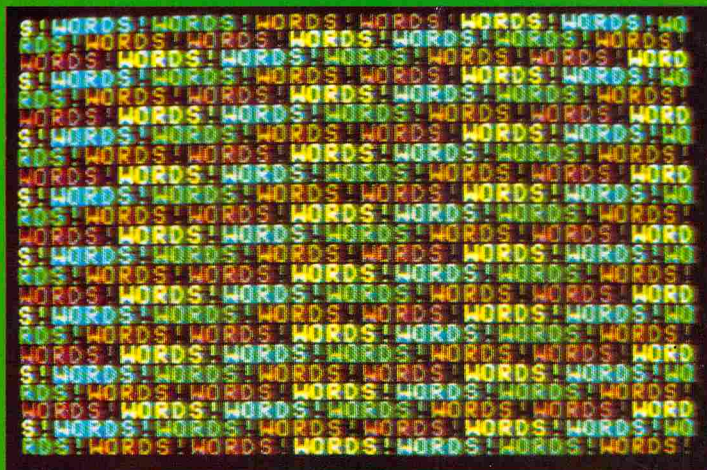


**MICRO
MATES**

Simple Words and Word Games



**PROGRAMS
FOR THE
COMMODORE 64
JONATHAN INGLIS**

JONATHAN INGLIS

Simple Words and Word Games

Photography
MARK GATEHOUSE

Consultant Editor
NAOMI BURGESS



GRANADA

Published by Granada Publishing 1985
Granada Publishing Ltd
8 Grafton Street, London W1X 3LA

Copyright © Jonathan Inglis 1985
Illustrations copyright © Granada Publishing 1985

British Library Cataloguing in Publication Data
Inglis, Jonathan

Simple words and word games. – (Micro mates for
the Commodore 64; 3)

1. Computer games 2. Commodore 64 (Computer)
– Programming

I. Title II. Series

794.8'028'5404 GV1469.2

ISBN 0-246-12603-5

Printed in Italy by
New Interlitho, Milan

All rights reserved. No part of this publication may
be reproduced, stored in a retrieval system, or
transmitted, in any form, or by any means, electronic,
mechanical, photocopying, recording or otherwise,
without the prior permission of the publishers.

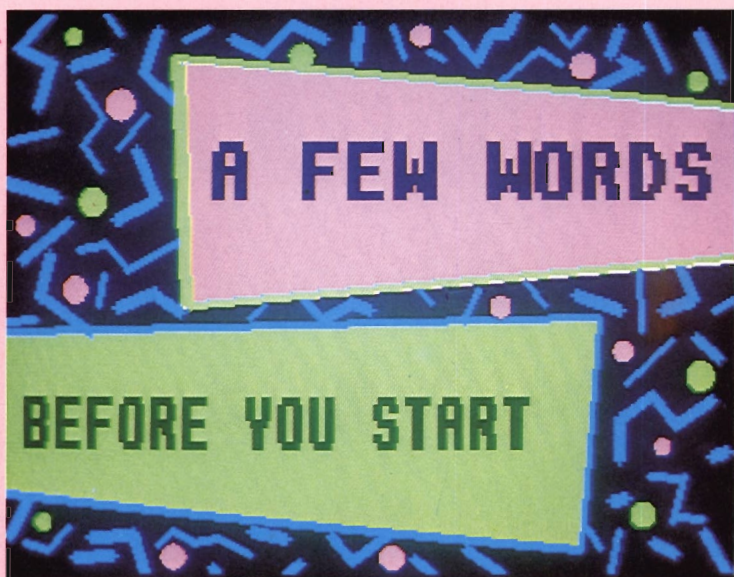
Simple Words and Word Games

This book is part of the MICRO MATES series. It will help you learn about BASIC programming on your Commodore 64. In it you will find a series of programs to type in and try out. They will show you how to write simple computer programs for yourself.

MICRO MATES programs are short, simple and fun to use. Every program is described and explained so that you can understand how they work and how you can change them.

Along with the program descriptions you will find lots of ideas: how to change your programs, how to make them run more efficiently and how to use all the short cuts and special features your Commodore 64 has to offer.

Although you could use this book if you've no experience of computers, you will probably come across some terms you don't understand. If so, you might like first of all to read my "Beginners' Micro Guide" on the Commodore 64, so that you have some idea how your computer works. It will make many ideas and words in this book easier to follow.



The programs in this book have been arranged so that you learn, step by step, the words and commands of BASIC used to write programs. Computer programs can be stored on cassette tape. The first few programs are simple introductions to some of the ideas that you will explore later. You may not want to 'save' these, but you will probably want to make a copy of some of the longer programs.

At the back of this book you will find a short section which tells you how to record programs. With the MICRO MATES series, you can build up a collection of your own programs on tape.

Before you start, read through this section carefully. It will help you to avoid some common problems. It may also help to keep a notepad and pencil handy for making notes.

LINE NUMBERS

A computer program is essentially a list of numbered instructions. Each set of instructions begins with a line number.

These usually start at 10 and go up in tens, though there may be exceptions. As the program grows, more lines will be added. So you should always type in the line numbers you are given.

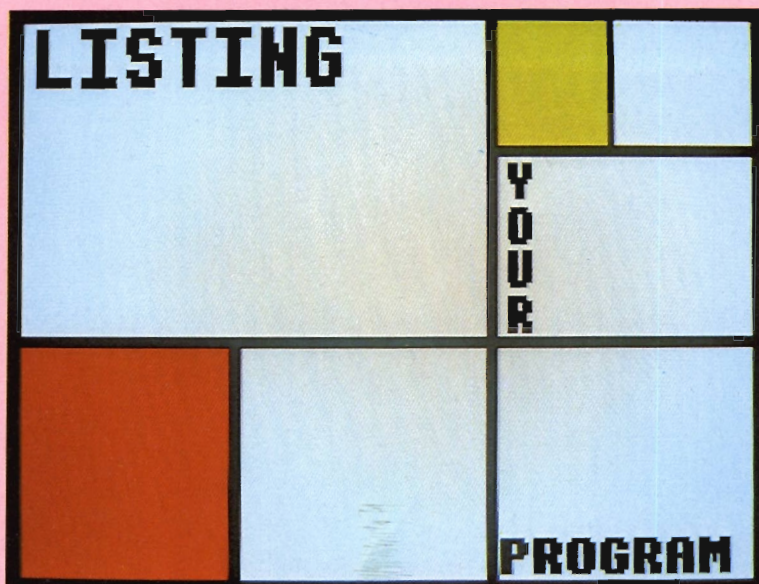
NOTE !

Sometimes the end of a line is carried over to the next line. It is still part of the same program line, however, unless it begins with a new line number. You should type it in as one line, and not press RETURN until you get to the end.

THE RETURN KEY

When you have typed in the line number and the instructions, you MUST press the RETURN key. If you just carry on typing in the next line number and set of instructions, your program will not work. By pressing the RETURN key, you tell the computer to store the line in its memory.





Once you have typed in a program you have to tell your Commodore 64 to 'run' it. To do this, type in RUN, then press RETURN. Once your program has run you may want to change it or look at it again. This is known as 'listing' your program.

To list a program, type in the command LIST and press RETURN. You will then be able to see your program. Where a program listing is too long to fit on the screen, it runs off the top of the screen. This is known as 'scrolling'. If you want to look at the first part of a program, the easiest way to stop it scrolling is to press the RUN STOP key.

You can also ask the computer to list just one part of a program. If you want to list, say, the first 100 lines of a program, type in:

```
LIST -100
```

(using the minus sign for the dash) and press RETURN. This will list the first 100 lines

only. This is very helpful with longer programs. If you want to see, for example, Lines 100 to 150, then type in:

LIST 100-150

and press RETURN. Individual lines can also be listed. For example, if you want to see Line 120 of a program, type in:

LIST 120

and press RETURN. This can be very useful when you think you may have made a mistake on a particular line of a program.

PROGRAMMING FOR BEGINNERS

Your computer works by obeying instructions written in the language called BASIC, and BASIC has some very strict rules. If you don't follow these, your programs will crash. This may be because the program you've typed in contains a 'syntax error'.

When you type in a computer program, you have to copy it in correctly. Even little things, such as not leaving a space between words or leaving too much space between certain symbols, can cause your programs to crash. Watch out for those punctuation marks, they are more important than they look. The semi-colon symbol ; has quite a different meaning from the colon symbol : and they should not be confused with each other.

If all goes well, your program will work first time. Unfortunately things don't always happen quite so smoothly and your program may not work properly.

When this happens your computer will probably tell you that you have made an 'error'.

HUNT THE BUG

Sometimes it will also tell you on what line the error has been made. You will then need to list the program or the line or lines concerned, and find the error.

If your Commodore 64 tells you on which line the mistake has been made, compare this line in your listing with the same line in the listing in the book. Sometimes the mistake (or 'bug' as they are known in computing) is obvious. When you have found it, rewrite the line, altering the offending word or section.

ZEROS AND O'S

A common mistake made by beginners is to confuse the 0 (zero) with the O (letter O). Make sure that you use the zero 0 when typing in numbers. Your Commodore 64 will not accept an 'O' for a '0'.

CORRECTING MISTAKES

If you notice while you are typing in a line that you have made a mistake, you can use the INST DEL key to correct it. Pressing this key rubs out the last letters or numbers that you have typed in. When you have done this, just type in the correct instructions.

If you notice a mistake in an earlier line, there are two things you can do.

One way of correcting it is to type in that line again with the right instructions, and press RETURN. The new version automatically replaces the old one.

Another way to alter your programs is to use the 'screen editor'.

USING THE SCREEN EDITOR

You can use the screen editor to change a line, to add to it, or to renumber it.

Changing a line

Type in the following program:

```
10 PRINT "      HELLO"  
20 PRINT "!.....!"  
30 PRINT "MY NAME IS FRED"
```

Now press the SHIFT key and, at the same time, the CRSR key with arrows going up and down. This moves a block (called the cursor) up the screen. Position the cursor so that it is on the 1 in Line 10.

Now press the CRSR key with arrows pointing left and right. This moves the cursor along the line. Move the cursor so that it is on the second L and type in the letter P. Next press the space bar to remove the letter O,

and finally press the INST DEL key to remove the space at the end of the string.

Press the RETURN key and then the left hand CRSR key (the one with arrows going up and down), until the cursor block is positioned below the listing. Now list the program, and you will see that Line 10 has been changed.

How to add to a line

It is just as easy to add to a line with the screen editor. Move the cursor up the screen to the 3 in Line 30. Now move it along until it is positioned on the space after IS. Press the SHIFT and INST DEL keys at the same time, four times. This opens up a gap in the program line.

Type in the word NOT, press RETURN and move the cursor below the program. LIST the program, and you will see the difference.

How to renumber a line

Now move the cursor to the 2 in Line 20, type in the number 9, and press RETURN. When you list your program this time, you will see that as well as Line 20 you also have a Line 90 which contains the same instructions.

This can be very useful where you have a program containing several lines with identical instructions. Just type in the first of those lines in full; after that, all you need to do is change the line numbers.

Pressing the left hand CRSR key without SHIFT moves the cursor down, and pressing the right hand CRSR key with SHIFT moves it to the left.

Make sure that you always press the RETURN key when you have finished, to store the newly edited line in the computer's memory. If you use the RETURN key to move the

cursor to a position below the program, you will see the message

ROUT OF DATA ERROR

You can ignore this - it doesn't mean you've made a mistake.

GETTING RID OF A PROGRAM

If you want to start again, or move on to another program, you should remove your old program from the computer's memory. To do this, type in the command:

NEW

and then press RETURN. But don't do this until you're sure you are finished with the old program. Once a program has been removed, you can't get it back again.

ABBREVIATIONS

You don't always need to type BASIC words in full. For example, the command LIST can be shortened by pressing L and then SHIFT and I together.

Many BASIC words can be abbreviated without affecting the program listings. A full list of the abbreviations that you can use is in your User Manual, on pages 130 and 131.

CLEARING UP

When you are typing in or running programs, your screen can get rather cluttered. If you are using colour, the program listings can also become hard to read.

To clear the screen, press the SHIFT and CLEAR HOME keys at the same time.

To restore the screen display to the same colours as when you switch on, press the RUN STOP and RESTORE keys at the same time. You will need to tap the RESTORE key sharply.

THE SHIFT LOCK KEY

If the keyboard isn't responding as you want it to, it may be because you have accidentally switched the SHIFT LOCK key on. The SHIFT LOCK key is off when it is up.

THE COMMODORE KEY

At the bottom left hand corner of your keyboard you will see a key with the Commodore trademark printed on it. This is the 'Commodore key'. It is used to obtain some of the control codes and graphics shapes.

GRAPHICS SHAPES

On the front edge of some of the keys on your keyboard you will see a series of shapes and symbols known as 'graphics shapes' or 'characters'. These can be printed on the screen.



















To obtain the shapes on the left hand side of the keys, press the Commodore key and the shape key at the same time.

To obtain the shapes on the right hand side of the keys, press the SHIFT key and the shape key at the same time.

COLOUR CONTROL CODES

In the programs in this book you will see that some of the listings contain strange graphics shapes. These are colour control codes. They can be used to change the colour of the text that you put on screen.

Look at the table below. It shows the codes available and which keys to press to obtain them. For example, to get orange writing you should press the Commodore key and key 1 at the same time. The REVERSE ON code simply reverses the normal foreground and background colour of the letter. To get back to normal letters on the same line, use the REVERSE OFF code.

CMDR	1		ORANGE	CTRL	1		BLACK
CMDR	2		BROWN	CTRL	2		WHITE
CMDR	3		LIGHT RED	CTRL	3		RED
CMDR	4		GREY 1	CTRL	4		CYAN
CMDR	5		GREY 2	CTRL	5		PURPLE
CMDR	6		LIGHT GREEN	CTRL	6		GREEN
CMDR	7		LIGHT BLUE	CTRL	7		BLUE
CMDR	8		GREY 3	CTRL	8		YELLOW
				CTRL	9		REVERSE ON
				CTRL	10		REVERSE OFF

IF ALL ELSE FAILS ...

For all sorts of reasons a computer sometimes 'locks', that is, it refuses to work. If this happens, you will just have to switch off and start again.

THE COMPUTER'S WAY WITH WORDS

Many people think that computers can only be used to add up numbers, or work out complicated sums. Although they can be used like this, computers can also be used with words, to write letters, store information and even play word games.

At the same time your computer doesn't really understand English, it understands BASIC, a special language for computer programs. So you need to learn a few BASIC rules before you start.

These rules are rather like the rules of punctuation. You have to remember to use the right symbols, otherwise your computer won't work properly. They are known as the 'syntax' of computing. If you make a mistake, your computer will tell you that you have made a 'syntax error'.

Many simple syntax errors are caused by people forgetting those all-important BASIC symbols.

When you use a computer for the first time, you may not be used to typing. Part of the fun of using your computer is learning how to use a typewriter-style keyboard.

The first few programs in this book are designed to give you practice at using the keyboard.

PROGRAM 1: GETTING YOUR MESSAGE ACROSS

Program 1 is very simple. It shows you one or two things about BASIC and words, and about controlling the screen display.

Line 10 uses the SHIFT/CLR HOME symbol. To print this shape, type in PRINT" and then press the SHIFT and CLR HOME keys at the same time. This line clears the screen.

Line 20 then uses the instruction POKE 53280 to change the colour of the screen border. The 1 which follows stands for the colour white. POKE 53281 then changes the colour of the background. (You can easily remember which is which if you think of the 0 as O for Outside, and the 1 as I for Inside.)

Line 30 uses a variable to store the word COMMODORE. The variable name is N\$. The \$ sign is very important. Your Commodore 64 cannot tell the difference between words and numbers. When you want to store a word or group of words, that \$ sign must follow the variable name.

The word or name COMMODORE has quotation marks around it. This is another example of BASIC syntax (the rules of the language). When you want to put words in a program you must use quotation marks, although in Commodore BASIC you can leave out the second set if you wish.

At Line 40 the BASIC word PRINT tells your Commodore 64 to print whatever words, numbers etc follow on that line.

Instead of saying COMMODORE, it says N\$ because N\$ now stands for COMMODORE. The rest of Line 40, the words " IS FANTASTIC", will also be printed out on the screen when the program is run.

Type in:

```
10 PRINT "C"  
20 POKE 53280,1:POKE 53281,1  
30 LET N$="COMMODORE"  
40 PRINT N$ " IS FANTASTIC"
```

PRINTING SPACES

Your Commodore 64 prints out exactly what it finds inside the quotation marks. In Program 1 a space was left before IS, inside the quotes at Line 40. If you left it out, the message would appear on the screen like this:

COMMODOREIS FANTASTIC

So you should always leave a space where it is needed. If you see a space left inside quotes in a program listing, make sure you type it in.

The SPACE key is the long one at the bottom of the keyboard.

YOUR NAME IN LIGHTS!

At Line 30, why not put your own name, instead of the word COMMODORE? Or, why not change the message at Line 40? Try writing your own version of this program so that you get used to using those variables and quotation marks.

Remember, you can also change the colour of your displays by changing the numbers after POKE 53280 and POKE 53281, at Line 20.

After you've done that, the colours might make your screen display difficult to read. To get a listing in the standard screen display colours, hold down the RUN STOP key and tap the RESTORE key. Then just type in the command LIST and press RETURN.

PROGRAM 2: FRIENDLY WELCOME

Program 2 makes an excellent start to any program you may write. It asks you your name, then says 'HELLO' to you.

Line 20 clears the screen with the SHIFT/CLR HOME instruction. Line 30 prints the question 'WHAT IS YOUR NAME?'.

At Line 40 the word INPUT is used; this allows you to 'put in' your name.

When you run this program you will see the question printed out. On the next line you will also see a question mark followed by a flashing block (the cursor). Type in your name and press RETURN, and your name will be stored in the computer's memory.

Your name is stored as the variable NAME\$. Notice that \$ sign again! You could just use N\$ here, but NAME\$ is clearer. It is helpful to use variable names that tell you what the variable stands for.

Line 60 changes the colour of the screen display. Line 70 then prints the message 'HELLO ', followed by NAME\$.

Just to show that the computer hasn't forgotten your name, Line 90 prints another message to you. Make sure you type in the whole line before pressing RETURN.

Line 80 uses the word PRINT on its own. This leaves a blank line between the printed messages and makes the screen display more attractive. Always try to make your screen displays easy to read.

USING COLOUR

You can change the colour in which words are printed on the screen, too.

For example, to print the letters in Program 2 in purple, retype Line 30 - and just before you type in the word WHAT, press the CTRL key and key 5 at the same time. This prints the symbol or CONTROL CODE for purple. Type the rest of the line in. When you run the program, your letters will be purple.

On the front of keys 1 to 8 are the other colours you can get in this way. With CTRL and key 1, for example, you will get BLK=black letters. Each colour has its own control code or symbol for listings, but these symbols do not get printed on the screen when the program runs.

Type in:

```
10 REM * FRIENDLY START *
20 PRINT " "
30 PRINT "WHAT IS YOUR NAME ?"
40 INPUT NAME$
50 PRINT " "
60 POKE 53280,6:POKE 53281,6
70 PRINT "HELLO " NAME$
80 PRINT
90 PRINT "I HOPE YOU ENJOY THIS
PROGRAM, " NAME$
```

THE REM STATEMENT

At Line 10 you will see what is known as a REM statement. REM stands for REMARK. A REM statement tells you what a program is about and can be very useful.

Although you can leave them out if you want, it is considered bad manners in computing not to use REM statements. It isn't always easy to know what a program does, just by looking at a listing. REM statements can help you find out.

PROGRAM 3: FACT FILE

Program 3 asks several questions and then prints the answers out in an official looking way.

Lines 30 to 100 ask you for certain items of information. Lines 110 to 190 then print out all that information in a more interesting fashion. The underlining at Lines 130, 150, 170 and 190 is done by typing in the minus sign.

```
What is your name ?  
? Jason Brown  
How old are you ?  
? 13  
Date of birth ?  
? 6th July 1971  
What school do you go to ?  
? Bright Side Avenue
```

You can easily adapt this program to suit your own interests. You could ask, for example, about a person's favourite pop group, football team, colour or food.

Type in:

```
10 REM * FACT FILE *  
20 PRINT "Q"  
30 PRINT "WHAT IS YOUR NAME?"  
40 INPUT NAME$  
50 PRINT "HOW OLD ARE YOU?"  
60 INPUT AGE$  
70 PRINT "DATE OF BIRTH?"  
80 INPUT BIRTHDAY$  
90 PRINT "WHAT SCHOOL DO YOU GO TO?"  
100 INPUT SCHOOL$  
110 PRINT "Q":POKE 53280,0:POKE 53281,0  
120 PRINT "NAME: " NAME$  
130 PRINT "_____  
140 PRINT "AGE: " AGE$  
150 PRINT "_____  
160 PRINT "DATE OF BIRTH: " BIRTHDAY$  
170 PRINT "_____  
180 PRINT "SCHOOL: " SCHOOL$  
190 PRINT "_____
```


UPPER AND LOWER CASE LETTERS

One way of improving your screen displays is to use upper and lower case, that is, capital and small letters. If you would like to do this, then (before you type in Program 3) press the Commodore and SHIFT keys at the same time. This changes all capital letters to small ones.

Type in your program, using small letters for the BASIC programming words such as INPUT or PRINT too. Where you want a letter to be printed as a capital, press SHIFT and that letter.

Now type in the command run (not RUN). Then type in your answers, using small letters or capitals as needed. The information you put in will be printed out in upper and lower case letters.

If you want to return to capital letters, press the Commodore and SHIFT keys at the same time again. Try this and see what it does to your listing, and to the program when you run it.

STRINGS

You can put anything you like in quote marks, not just letters, words and spaces - except another set of quote marks. For example, you can put in numbers, shapes, or graphics characters.

A set of information placed inside quote marks is called a 'string', because the items of information (whatever they may be) are 'strung' together like beads on a thread. The quote marks are rather like knots at each end of the string.

PROGRAM 4: PASSWORD

Program 4 asks you for a password, and uses the words IF and THEN to make the computer decide whether your answer is correct. You can put these lines at the beginning of any of your programs.

Line 30 makes the variable PASSW\$ stand for COMMODORE. Line 40 asks you to give the password, and Line 50 INPUTs your answer.

Line 60 compares your answer, ANSWER\$, with the password PASSW\$. If your answer is the same as the password then Line 60 prints a welcoming message for you.

Line 70 tells the computer to see if your answer is NOT the same as the password, and if not, it makes the computer respond with a less friendly message.

Type in:

```
10 REM * PASSWORD *
20 PRINT "[C]"
30 LET PASSW$="COMMODORE"
40 PRINT "PLEASE GIVE THE SECRET
  PASSWORD"
50 INPUT ANSWER$
60 IF ANSWER$=PASSW$ THEN PRINT "WELCOME
  FRIEND"
70 IF NOT ANSWER$=PASSW$ THEN PRINT
  "ACCESS TO PROGRAM DENIED"
```

WARNING !

Anybody using Program 4 who doesn't know the password can easily find it out by typing in LIST. This lists the program. If they see Line 30, your secret is out!

PROGRAM 5: LOOPS AND LINES

This program shows you how to computerise a boring punishment!

It uses a FOR...NEXT loop. This is a way of making your Commodore 64 obey a set of instructions more than once.

You INPUT the number of lines at Line 40. Line 60 allows you to type in the line to be repeated, for example, 'I MUST NOT TALK IN CLASS'.

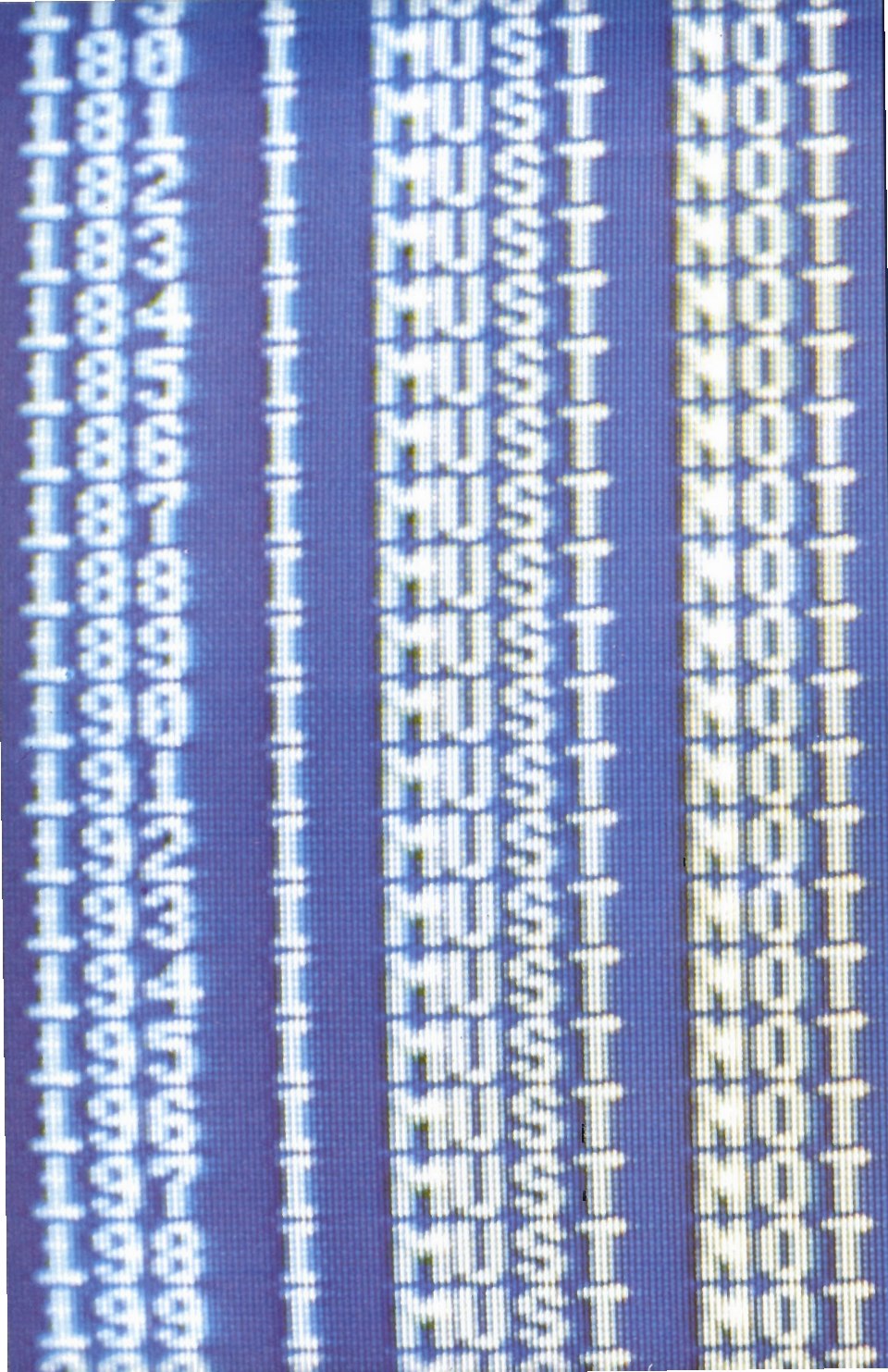
Line 90 sets up the loop to run for as long as you set at Line 40. Line 90 prints the number of your line and the line itself. It will go on doing this until it has printed the right number of lines.

Line 100 slows the program down a little, using a FOR...NEXT loop to waste time.

LONGER VARIABLE NAMES

Many programs in this book use longer variable names. When you write your own programs, choose your longer variable names carefully! Make sure you never use two with the same first two letters. Your Commodore 64 only looks at the first two letters of each variable name, so it would not be able to distinguish them.

Also, if the first two letters of a variable name are the same as those of a BASIC word, your Commodore 64 may not accept them.



Type in:

```
10 REM * LINES *  
20 PRINT "C"  
30 PRINT "HOW MANY LINES HAVE YOU BEEN  
   GIVEN?"  
40 INPUT NUMBER  
50 PRINT "TYPE IN THE LINE YOU HAVE  
   BEEN GIVEN"  
60 INPUT LINE$  
70 PRINT "C"  
80 FOR N=1 TO NUMBER  
90 PRINT N; LINE$  
100 FOR P=1 TO 100: NEXT P  
110 NEXT N
```

PROGRAM 6: WORD PATTERN

Program 6 fills the screen with whatever message you choose.

Lines 20 to 50 ask for your message and the number of times you want to see it printed on the screen.

Line 70 sets the background and border colours. Line 80 shows a different method of changing the colour of any letters or text printed on the screen. The instruction `PRINT CHR$(31)` has the same effect as using the control code for blue. A full list of these and other codes can be found in Appendix F of the User Manual.

Line 90 sets up the loop to repeat the instructions at Line 100. Notice the use of the semi-colon `;` here. This tells your Commodore 64 to print the next piece of text next to the last piece, not on the following line. Try leaving this out to see what happens.

You can type in any message you like. Different messages produce different patterns.

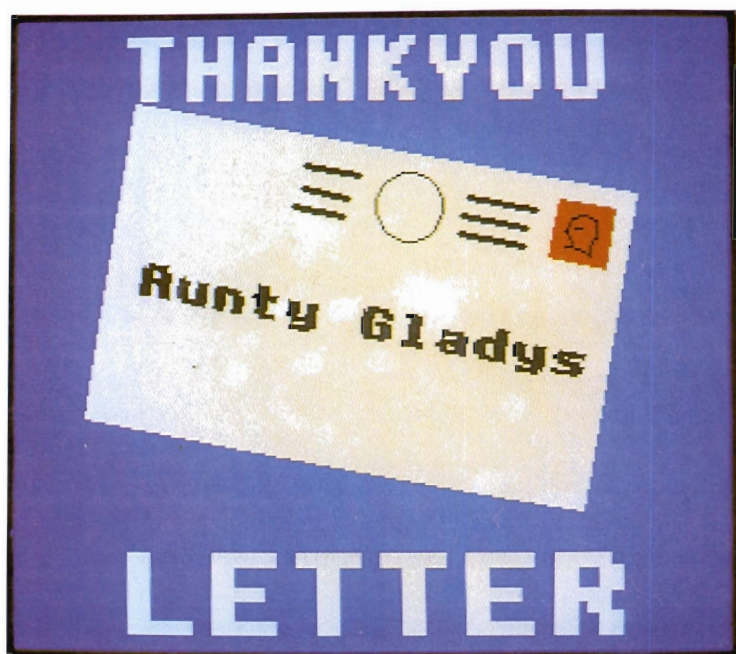
Type in:

```
10 REM * WORD PATTERN *
20 PRINT "WHAT IS YOUR MESSAGE?"
30 INPUT MESSAGE$
40 PRINT "HOW MANY TIMES DO YOU WANT TO
   SEE IT?"
50 INPUT NUMBER
60 PRINT "█"
70 POKE 53280,8:POKE 53281,8
80 PRINT CHR$(31);
90 FOR N=1 TO NUMBER
100 PRINT MESSAGE$;
110 NEXT N
```


ADDING SOUND

If you would like to make Program 6 more fun by adding an interesting sound effect, type in the following lines:

```
92 POKE 54276,0:POKE 54273,0
94 POKE 54296,15:POKE 54277,127
96 POKE 54276,17:POKE 54273,91
98 FOR P=1 TO 30:NEXT P
120 POKE 54276,0:POKE 54273,0:POKE
    54277,0
```

PROGRAM 7: THANK YOU NOTE

Program 7 helps you to write computerised thank you letters!

The program asks you for certain items of information such as who the letter is for, and what present they gave you.

Line 70 allows you to include a special message to make the letter more personal.

Finally it asks you who you are so that you can 'sign' the letter.

You can easily change this program to suit your own interests and needs. Try adapting it to write other kinds of letter, for example Christmas greetings or an invitation to your next birthday party.

Type in:

```
10 REM * THANK YOU LETTER *
20 PRINT "C"
30 PRINT "WHO IS YOUR LETTER FOR?"
40 INPUT WHO$
50 PRINT "WHAT PRESENT DID THEY GIVE
   YOU?"
60 INPUT PRESENT$
70 PRINT "TYPE IN A SPECIAL MESSAGE FOR
   THEM"
80 INPUT MESSAGE$
90 PRINT "WHAT IS YOUR NAME?"
100 INPUT NAME$
110 POKE 53280,1:POKE 53281,1
120 PRINT "C"
130 PRINT "DEAR " WHO$ ", "
140 PRINT "THANK YOU FOR THE LOVELY "
   PRESENT$ ". "
150 PRINT "IT WAS JUST WHAT I WANTED,
   AND"
160 PRINT "I AM EVER SO PLEASED WITH
   IT."
170 PRINT: PRINT MESSAGE$
180 PRINT: PRINT "LOTS OF LOVE,"
190 PRINT: PRINT NAME$
```

YES/NO ROUTINE

Now here is a useful 'routine'. A routine is a group of lines that can be added on to other programs.

Once you've run your program, this routine asks you if you want another go. If so, type in YES and it sends the program back to the beginning. If not, type in NO and the program ends.

This program uses Line 200 as its starting point but any line after the end of your program will do.

At Line 230, where it says GOTO, you should put in the number of the line with which your main program begins. With Program 7, for example, you would put in 20.

Type in:

```
200 REM * YES/NO ROUTINE *  
210 PRINT "WOULD YOU LIKE ANOTHER GO,  
    YES OR NO?"  
220 INPUT ANSWER$  
230 IF ANSWER$="YES"THEN GOTO 20:END
```

PROGRAM 8: STORYTELLER

Program 8 is similar to the thank you note program, except that it tells a story.

Line 40 uses the control code for REVERSE ON. To obtain this, press the CTRL key and key 9 at the same time. This prints the program's title in reversed colours and makes an attractive heading.

Lines 50 to 120 ask you for various pieces of information about the story.

Lines 60, 70 and 80 use INPUT to print the question as well as to INPUT the answer. This method avoids the need for a separate PRINT line, making it quicker to program questions.

Lines 130 to 160 ask you if you'd like to see the story. Lines 180 to 280 then tell you the story.

Although this a long program, it is really quite a simple one. Try adapting it to make up a story of your own. There's no need for the program to stop at Line 280 - you can make it much longer if you like. You could also improve it by using upper and lower case letters.



You may want to keep a taped copy of this program, so that you can use it again without having to type it in each time.

Type in:

```
10 REM * STORYTELLER *
20 PRINT "G"
30 POKE 53280,7:POKE 53281,7
40 PRINT " B STORYTELLER PROGRAM "
50 PRINT:INPUT "WHAT IS THE NAME OF THE
  PERSON IN THE STORY ";NAME$
60 INPUT "WHERE DOES "NAME$" LIVE ";
  PLACE$
70 INPUT "WHO ARE THEY AFRAID OF ";
  ENEMY$
80 INPUT "WHO IS THEIR FAVOURITE
  PERSON ";FAV$
90 PRINT "WHAT PRESENT WOULD YOU GIVE "
  FAV$ "?":INPUT PRESENT$
```

Listing continues on next page

```

100 PRINT "WHAT LESSON DOES " NAME$
    " NOT LIKE?":INPUT SCHOOL$
110 PRINT "WHAT IS YOUR NAME?"
120 INPUT YOU$
130 PRINT "WOULD YOU LIKE TO SEE THE
    STORY, Y OR N?"
140 INPUT ANSWER$
150 IF ANSWER$ ="Y" THEN GOTO 170
160 IF NOT ANSWER$ ="Y" THEN END
170 PRINT "█"
180 PRINT "ONCE UPON A TIME, THERE WAS
    SOMEBODY CALLED " NAME$"."
190 PRINT NAME$ " LIKED LIVING IN "
    PLACE$ "."
200 PRINT "BUT " ENEMY$ " WAS REVOLTING,
    AND " NAME$ " WAS VERY AFRAID OF "
    ENEMY$ "."
210 PRINT FAV$ ", WHO WAS A FRIEND OF "
    NAME$ ", KNEW THIS."
220 PRINT "SO ONE DAY, " FAV$ " POURED
    PAINT ALL OVER " ENEMY$ "."
230 PRINT NAME$ " WAS SO PLEASED THAT
    NAME$ " GAVE " FAV$
240 PRINT " THEIR FAVOURITE " PRESENT$
    "."
250 PRINT NAME$ " WAS SO HAPPY THAT EVEN
    GOING TO "
260 PRINT SCHOOL$ " SEEMED FUN THAT DAY."
270 PRINT:PRINT "THIS STORY WAS WRITTEN
    BY "
280 PRINT YOU$

```

PROGRAM 9: HOW MANY LETTERS

Program 9 will tell you how many letters there are in any word. This program uses the BASIC word LEN.

Line 30 asks you to type in a word. You can type anything you like - try it with your name.

Line 40 then works out how many letters there are in your word, using the word LEN. This number is stored as the variable NUM. Lines 50 and 60 then print out the answer.

Type in:

```
10 REM * LETTER COUNT *  
20 PRINT "C"  
30 INPUT "TYPE IN YOUR WORD ";W$  
40 LET NUM=LEN(W$)  
50 PRINT "THERE ARE " NUM " LETTERS  
60 PRINT "IN THE WORD " W$
```

PROGRAM 10: BITS AND SLICES

Program 10 shows you how to use the BASIC word MID\$ to print parts of a string or message.



Line 30 sets the variable W\$ to stand for the word "DISCONTENTED".

Line 40 uses MID\$ to print the first four letters at the beginning of W\$. The first number in brackets says at which letter to begin printing, the second says how many letters are to be printed from that point.

Line 50 uses MID\$ to print CONTENT, the seven letters including and following the letter C. Line 60 prints the word 'TENT'.

Type in:

```
10 REM * USE OF MID$ *  
20 PRINT "[C]"  
30 LET W$="DISCONTENTED"  
40 PRINT MID$(W$,1,4)  
50 PRINT MID$(W$,4,7)  
60 PRINT MID$(W$,7,4)
```

Can you add more lines to this program, to print different bits (sometimes called 'slices') of the word DISCONTENTED ?

PROGRAM 11: BACK TO FRONT

Program 11 lets you type in a message, and prints that message in reverse. Although it is rather a short program, it is actually quite complicated.

It uses the BASIC word MID\$ to break your message down into individual letters, then it prints these in reverse order.

Line 60 works out how many letters there are in your message, using LEN. The answer is stored as the variable NUM. Line 70 sets up a loop to count down from NUM to 1.

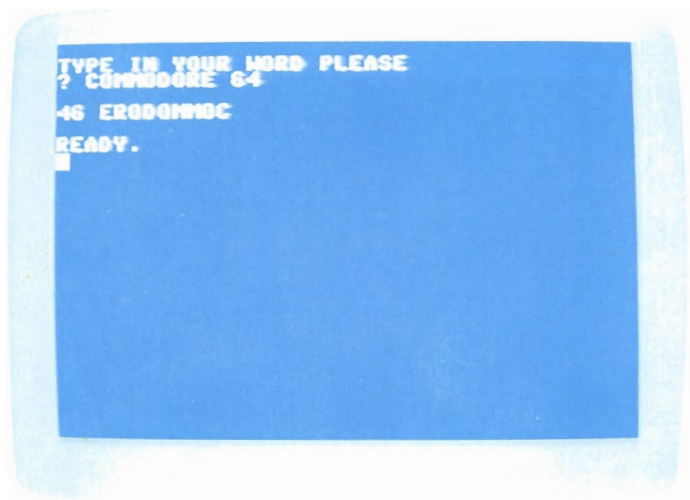


Line 80 then selects each letter of your word, one at a time, starting at the end, and prints it.

Line 90 closes the loop and sends the program back to Line 70, so that the next letter can be picked.

Type in:

```
10 REM * BACK TO FRONT *
20 PRINT "C"
30 PRINT "TYPE IN YOUR WORD PLEASE"
40 INPUT W$
50 PRINT
60 LET NUM=LEN(W$)
70 FOR L=NUM TO 1 STEP -1
80 PRINT MID$(W$,L,1);
90 NEXT L
100 PRINT
```



PROGRAM 12: READ & DATA

Program 12 shows you how the BASIC words READ and DATA are used. They offer an easier way of storing information than with LET statements.

The information to be printed is stored in DATA statements at Lines 70 to 90.

Line 30 'reads' the information stored in the DATA statements. Notice the order in which the DATA is arranged. The first variable at Line 30, NAME#, matches up with the first item in the DATA statement at Line 70. The second variable NUMBER# matches the item in the second DATA statement at Line 80.

Lines 40 to 60 then take the information or data that has been read and print it out.

By changing the information in the data statements you can change the name, number and address printed by this program.

SAVING TIME AND SPACE

In Commodore BASIC you do not need to use the word LET. In Program 11, for example, Line 60 could read:

```
60 NUM=LEN(W$)
```

You can also type in the ? symbol, instead of the word PRINT. So, if you type in this:

```
10 ? "HELLO"
```

it has exactly the same effect as if you typed in this:

```
10 PRINT "HELLO"
```

Type in:

```
10 REM * READ and DATA *
20 PRINT "[C]"
30 READ NAME$,NUMBER$,ADDRESS$
40 PRINT NAME$
50 PRINT NUMBER$
60 PRINT ADDRESS$
70 DATA MIKE ROE
80 DATA 64K
90 DATA MEMORY LANE
```

PROGRAM 13: SPELLING TEST

Program 13 uses READ and DATA to display four sets of words. In each set one word is spelt wrong. Whoever is using the program has to type in the word they think is incorrect.

Line 40 sets up the loop so that the program asks you a question four times.

```
code symbol softwear micro
```

```
Which of these words is spelt wrongly ?
```

```
? ■
```

Line 100 checks the answer. If the answer is correct, you are congratulated. Line 110 checks in case the wrong answer has been given, and tells you if it has.

Finally Lines 200 to 240 list the words including, at the end of each line, the incorrectly spelt word. Notice that the group of words is stored as one string, not separated by commas.

Notice too the use of control codes to make the screen display more interesting. The PRINT statements at Lines 60, 70 and 90 also help to make the layout of the screen display more attractive.

Type in:

```
10 REM * SPELLING TEST *  
20 PRINT "■"  
30 POKE 53280,5:POKE 53281,5  
40 FOR TEST=1 TO 4: PRINT "■"  
50 READ W$,N$  
60 PRINT:PRINT W$  
70 PRINT:PRINT"■"
```

```

80 PRINT "WHICH OF THESE WORDS IS SPELT
   WRONG?"
90 PRINT "■":INPUT WG$:PRINT "■"
100 IF WG$=N$ THEN PRINT "WELL DONE,
   THAT IS CORRECT":GOTO 120
110 IF NOT WG$=N$ THEN PRINT " NO,
   SORRY, WRONG ANSWER"
120 FOR PAUSE=1 TO 1000: NEXT PAUSE
130 PRINT "■":NEXT TEST
140 PRINT "■QUIZ OVER. THANK YOU."
200 DATA graphics loop statement
   comand,comand
210 DATA code symbol softwear
   micro,softwear
220 DATA print varieble input
   poke,varieble
230 DATA computer listing memory
   cassette, computer

```

Try adding more DATA lines, using hard words to quiz your friends with. Try and find some unusual ones in a dictionary. Remember to put a comma between the set of words and the last, wrongly spelt word. Finally, change the size of your loop at Line 40 to take account of the new number of questions.

PROGRAM 14: WORD ORDER

Program 14 is very similar to Program 13. It displays a group of words and asks you to rearrange them so that they make up a sentence.

Line 40 sets the size of your loop. If you want to add more questions, just increase the size of the loop. Remember that you must have enough information in your DATA statements to cover the number of questions being asked.

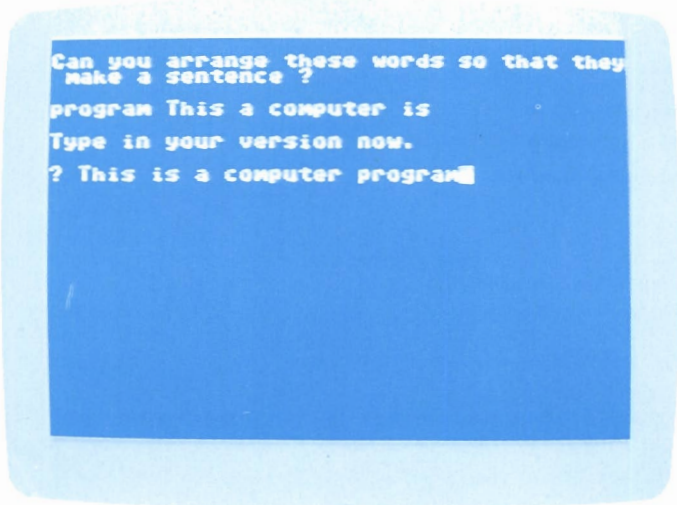
If you want to run this program several times, why not add the YES/NO routine? First, remove the final END statement from Line 150. The routine can then be put at Lines 152, 154 and 156.

Lines 40, 70, 80, 100, 110, 160 and 180 use PRINT CHR\$ statements to change the text colour. This is the same as using the control codes, but it makes your listings easier to read.

The loop at Line 140 just wastes time so that your screen display isn't immediately cleared.

The subroutine at Lines 160 to 190 prints the correct version of the sentence if you get the answer wrong.

This time, the information in the data statements is enclosed in quote marks. This is so that the computer will accept the words in capitals even if they look like BASIC words.



```
Can you arrange these words so that they  
make a sentence ?  
program This a computer is  
Type in your version now.  
? This is a computer program
```

Can you arrange these words so that they
make a sentence ?

a language BASIC is computer

Type in your version now.

? ■

Type in:

```
10 REM * WORD ORDER *
20 PRINT "■"
30 POKE 53280,13:POKE 53281,13
40 FOR TEST=1 TO 4:PRINT CHR$(30)
50 READ W$,SENT$
60 PRINT "CAN YOU ARRANGE THESE WORDS
   SO THAT THEY MAKE A SENTENCE?"
70 PRINT:PRINT CHR$(156):PRINT W$
80 PRINT:PRINT CHR$(30)
90 PRINT "TYPE IN YOUR VERSION NOW"
100 PRINT:PRINT CHR$(156):INPUT YR$
110 PRINT CHR$(156)
120 IF YR$=SENT$ THEN PRINT "CORRECT,
   WELL DONE"
130 IF NOT YR$=SENT$ THEN PRINT "SORRY,
   WRONG ANSWER.":GOSUB 160
140 FOR PAUSE=1 TO 2000:NEXT PAUSE
150 PRINT "■":NEXT TEST:END
160 PRINT:PRINT CHR$(30)
170 PRINT "THE CORRECT VERSION IS:"
180 PRINT:PRINT CHR$(156):PRINT SENT$
190 RETURN
```

Listing continues on next page

```
200 DATA "PROGRAM THIS A COMPUTER IS"
210 DATA "THIS IS A COMPUTER PROGRAM"
220 DATA "A LANGUAGE BASIC IS COMPUTER"
230 DATA "BASIC IS A COMPUTER LANGUAGE"
240 DATA "COMPUTER 64K HAS MEMORY THIS
    A"
250 DATA "THIS COMPUTER HAS A 64K
    MEMORY"
260 DATA "ON SAVED BE PROGRAMS TAPE CAN"
270 DATA "PROGRAMS CAN BE SAVED ON TAPE"
```

MULTI-STATEMENT LINES

Program 14, like many of the programs in this book, uses what are known as multi-statement lines. These are lines which contain a number of statements, separated by a colon : sign.

Line 150 of Program 14 is a good example of a multi-statement line.

Most (but not all) instructions and statements can be joined up in multi-statement lines. They save typing and program time, but be careful not to make your listings unreadable by using too many of them.

PROGRAM 15: NUMBERS AND LETTERS

Computers do not recognise letters as we do. In order to deal with letters, your Commodore 64 turns them into numbers.

Program 15 demonstrates the ASCII system (pronounced asskee). This is a standard system common to many computers. In the

ASCII system each letter has a code number; for example, the ASCII code for A is 65. In Program 15 a loop is set up to convert the numbers 65 to 90 into letters.

Line 40 uses the BASIC word CHR\$ to convert numbers into letters.

Type in:

```
10 REM * ASCII CODING *
20 PRINT "A"
30 FOR CODE=65 TO 90
40 PRINT CHR$(CODE);
50 NEXT CODE
60 PRINT
```

Notice how fast your Commodore 64 carries out this task. To slow it down, put in this line:

```
45 FOR P=1 TO 100:NEXT P
```

All the shapes and functions that can be typed in from the keyboard have their own ASCII code numbers.

PROGRAM 16: SECRET CODE

Program 16 turns any message you type in into code. It does this by taking each letter in your message and printing the next letter on in the alphabet.

The most important line in the program is Line 80. This adds one to the ASCII code number of each letter in your message. Line 100 then converts that code number back into a letter.

You can leave out Line 90 if you like. Then, where there were spaces in your first message, your coded message will print an exclamation mark.



WHAT IS YOUR SECRET MESSAGE ?
? I AM A COMMOORE 6-4

Type in:

```
10 REM * SECRET CODE *
20 PRINT "🔒"
30 PRINT "WHAT IS YOUR MESSAGE?"
40 INPUT MESSAGE$
50 LET NUM=LEN(MESSAGE$)
60 FOR N=1 TO NUM
70 LET C$=MID$(MESSAGE$,N,1)
80 LET CODE=ASC(C$)+1
90 IF CODE=33 THEN PRINT CHR$(32);:
    GOTO 110
100 PRINT CHR$(CODE);
110 NEXT N
120 PRINT:PRINT
```

PROGRAM 17: USING AN ARRAY

Program 17 shows you how to use an ARRAY. An array is a list made up of similar items of information, which you want to store in one group.

HIEROGLYPHICS

You can use Program 16 to turn your messages into pictorial code by adding these lines.

```
45 POKE 53280,0:POKE 53281,0:PRINT "C"  
80 LET CODE=ASC(C$)+96  
90 IF CODE=128 THEN PRINT CHR$(32);:  
   GOTO 110
```

Your message will now look like a new kind of Egyptian hieroglyphics!

In Program 17 the information to be stored is the days of the week. Each day has the same variable name, WEEK\$. This is followed by a number in brackets. This number is known as the SUBSCRIPT of that variable.

For example the variable WEEK\$ with the subscript 1 is SUNDAY. WEEK\$(2) is MONDAY and so on. The advantage of this is that you can select a particular day by referring to its subscript.

Line 80 asks you to type in a number. Line 100 then prints out the variable whose subscript you chose.

When you want to use an array, you have to reserve memory space for it by using a 'DIM statement'. In a DIM statement the BASIC word DIM (for dimension) is followed by the variable name and then, in brackets, the number of items to be stored in that array.

This should be placed early on in the program, before you start to use the variables. In Program 17 the DIM statement can be found at Line 30.

Type in:

```
10 REM * USING DIM *
20 PRINT "📅"
30 DIM WEEK$(7)
40 WEEK$(1)="SUNDAY":WEEK$(2)="MONDAY"
50 WEEK$(3)="TUESDAY":WEEK$(4)=
  "WEDNESDAY"
60 WEEK$(5)="THURSDAY":WEEK$(6)="FRIDAY"
70 WEEK$(7)="SATURDAY"
80 INPUT "TYPE IN A NUMBER FROM 1 TO
  7 ";N
90 IF N>7 THEN GOTO 70
100 PRINT:PRINT "DAY " N " IS " WEEK$(N)
```

PROGRAM 18: BUBBLE SORT

Program 15 showed you how your Commodore 64 can convert letters into numbers. It can also use ASCII codes to sort lists of words into alphabetical order. Program 18 shows you how to make it do this. The data to be sorted is stored in an array.

The main lines of this program are Lines 80 to 150. These lines compare each name in the array A\$, contained in the DATA lines at the end of the program, with all the other names in the array. The > sign at Line 100 means 'greater than'.

If the value of one name on your list is greater than the one with which it is being compared, that name's subscript (the number in brackets) is made larger. This means that that name moves further down the list. The value of B, 66 in the ASCII code, is greater than A, which is 65. C, at 67, is greater still - and so on.

This way, the program gradually rearranges the subscripts of all the names in the array A\$, so that in the end the names are



picked and listed in alphabetical order. Although it happens extremely fast, your Commodore 64 carries out a great many comparisons and alterations to the order of its list in the time.

This kind of sort is known as a 'bubble sort' because the lighter words, that is, the ones with lower ASCII values, float gradually to the top. At the end the 'lightest' word is at the top and the 'heaviest' at the bottom. The rest are in descending order in between.

Type in:

```
10 REM * BUBBLE SORT *  
20 PRINT "☐"  
30 NUMBER=20  
40 DIM A$(NUMBER)
```

Listing continues on next page

```

50 FOR N=1 TO NUMBER
60 READ A$(N)
70 NEXT N
80 FOR J=1 TO NUMBER
90 FOR K=J TO 2 STEP -1
100 IF A$(K)>A$(K-1) THEN GOTO 150
110 T$=A$(K)
120 A$(K)=A$(K-1)
130 A$(K-1)=T$
140 NEXT K
150 NEXT J
160 FOR I=1 TO NUMBER
170 PRINT A$(I)
180 NEXT I
190 DATA DAVID,SANJAY,LOUISE,SCOTT
200 DATA SALLY,JANE,ANDREW,MALCOLM
210 DATA PATRICK,ASIF,TONY,SUSAN
220 DATA TINA,PAULINE,KAREN,WENDY
230 DATA TASLEEM,BRADLEY,RAY,COLIN

```

USING INPUT

Program 18 can be changed so that you can type in a list of names directly instead of storing them in the DATA statements at the end of your program.

To do this you must first of all remove Lines 190 to 230. Then add these new lines to the program:

```

30 INPUT "HOW MANY NAMES ON YOUR LIST ";
NUMBER
60 PRINT "TYPE IN NAME NUMBER ";N
65 INPUT A$(N)
155 PRINT "C";"HERE IS YOUR LIST:"

```

When you run this program you must now say, first, how many names are on your list. Then type in each name, and the program will sort them into alphabetical order.



```
ANDREW  
ASIF  
BRADLEY  
COLIN  
DAVID  
JANE  
KAREN  
LOUISE  
MALCOLM  
PATRICK  
PAULINE  
RAY  
SALLY  
SANJAY  
SCOTT  
SUSAN  
TASLEEM  
TINA  
TONY  
WENDY  
READY.
```

PROGRAM 19: USING GET

Program 19 is a simple program that shows you how to use a very useful word in Commodore BASIC, GET.

GET waits for you to press a key. You can choose what the computer will do if certain keys are pressed.

Line 20 uses GET A\$. This programs your computer to wait for you to press a key. Line 30 checks to see if no key has been pressed, in other words if A\$ contains nothing. Do not leave a space between the quotation marks here. If no key has been pressed, the program returns to Line 20.

If the key that has been pressed is key 1 then the message at Line 40 is printed out. Lines 50 and 60 are similar. Line 80 goes back to Line 20 to see what key is going to be pressed next.

If you press key S, Line 70 clears the screen and the program skips to Line 90.

Line 90 then waits for you to press any key. Once you do the program ends and the word READY will appear on the screen. This kind of line is very useful, for example, if you want to introduce a pause while a program is running.

Type in:

```
10 REM * USING GET *
20 GET A$
30 IF A$="" THEN GOTO 20
40 IF A$="1" THEN PRINT "1-GET YOUR GUN!"
50 IF A$="2" THEN PRINT "2-AIM IT TRUE!"
60 IF A$="3" THEN PRINT "3-NOT AT ME!"
70 IF A$="S" THEN PRINT "【】":GOTO 90
80 GOTO 20
90 GET A$:IF A$="" THEN 90
```

PROGRAM 20: SPEED TYPING!

Program 20 shows how you can use GET to impress your friends with your amazingly fast typing!

When you run the program, pressing any of the letter or number keys - it doesn't matter which - will print a message on your screen. (It will look as if you were typing it, really fast and correctly.)

The program uses a GOTO loop which reads in the data stored at Lines 110 to 150.

Line 60 waits for you to press a key and then prints a word in the message on the screen. It goes on doing this until it reaches the final !. Then the word RESTORE resets the program so that the same set of data can be used again. The instruction PRINT CHR\$(13) has the same effect as pressing the RETURN key, so that at the end of each message the program moves on to the next line of the screen.

THIS IS JUST TO SHOW YOU JUST HOW FAST A
TYPIST I HAVE BECOME USING MY COMMODORE
64 . I CAN NOW TYPE HUNDREDS OF WORDS A
MINUTE !

THIS IS JUST TO SHOW YOU JUST HOW FAST A
TYPIST I HAVE BECOME USING MY COMMODORE
64 . I CAN NOW TYPE HUNDREDS OF WORDS A
MINUTE !

THIS IS JUST TO SHOW YOU JUST HOW FAST A
TYPIST I HAVE BECOME USING MY COMMODORE
64 . I CAN NOW TYPE HUNDREDS OF WORDS A
MINUTE !

THIS IS JUST TO SHOW YOU JUST HOW FAST A
TYPIST I HAVE BECOME USING MY COMMODORE
64 . I CAN NOW TYPE HUNDREDS OF WORDS A
MINUTE !

THIS IS JUST TO SHOW YOU JUST HOW FAST A
TYPIST I HAVE BECOME USING MY COMMODORE
64 .

A program like this one was used in a recent film about a young computer whizz kid. The actor was not a fast typist but the film showed him typing happily away, apparently producing all sorts of information on the VDU screen!

Type in:

```
10 REM * SPEEDO *
20 PRINT "Q"
30 POKE 53280,14:POKE 53281,14:PRINT
   CHR$(31)
40 READ W$
50 IF W$="!" THEN RESTORE
60 GET A$
70 IF A$="" THEN 60
80 PRINT W$;
90 IF W$="!" THEN PRINT CHR$(13)
100 GOTO 40
110 DATA THIS ,IS ,JUST ,TO ,SHOW ,YOU ,
120 DATA HOW ,FAST ,A ,TYPIST ,I ,HAVE ,
130 DATA BECOME ,USING ,MY ,COMMODORE
   ,64. ,
140 DATA I ,CAN , NOW ,TYPE ,HUNDREDS
   ,OF ,
150 DATA WORDS ,A ,MINUTE ,!
```


PROGRAM 21: PROGRAM TITLE

An attractive title for your programs makes them look a great deal more professional.

Lines 30 and 80 print a row of asterisks across the screen. Any shape could be used here. Lines 40 and 70 print space between the two borders.

Line 50 gives the title the variable name A\$. Here again any title will do, provided that it is not longer than 40 characters.

Line 60 then uses TAB and LEN to position the program's title in the middle of the screen.

Finally Line 90 waits for you to press a key. The rest of your program would follow this line.

Type in:

```
10 REM * SIMPLE TITLE *
20 PRINT "C"
30 FOR N=0 TO 39:PRINT "*";:NEXT
40 FOR N=1 TO 9:PRINT:;NEXT
50 A$="HAUNTED HOUSE ADVENTURE"
60 PRINT TAB((40-LEN(A$))/2);A$
70 FOR N=1 TO 9:PRINT:;NEXT
80 FOR N=0 TO 39:PRINT "*";:NEXT
90 GET A$:IF A$="" THEN 90
```

PROGRAM 22: TITLE PAGE

Program 22 offers a brighter and more colourful title sequence.


Line 20 stores the title of the program, using the variable T\$. Line 30 stores the name of the writer of the program.



MICRO MATES PROGRAMS

8Y

COMMODORE 64



Line 40 then sends the program to the subroutine at Line 200. This prints a series of coloured stripes across the screen using the reversed space shape, stored as A\$. You could change the control codes to your own favourite colours.

Line 80 then uses TAB to print the program's title, and Line 140 prints the writer's name in the middle of the screen.

Line 160 then returns the program to Line 200 to print another series of stripes across the bottom of the screen. Finally, Line 170 uses GET so that the program will not run on until a key is pressed.

Type in:

```
10 REM * TITLE PAGE *
20 LET T$="DIZZY TOWERS"
30 LET N$="EMMA CARTER"
40 PRINT "  ";GOSUB 200
50 PRINT:PRINT
60 LET L0=LEN(T$)
70 LET T=(40-L0)/2
80 PRINT TAB(T);T$
90 PRINT:PRINT
100 PRINT TAB(19);"BY"
110 PRINT:PRINT
120 LET L0=LEN(N$)
130 LET T=(40-L0)/2
140 PRINT TAB(T);N$
150 PRINT
160 GOSUB 200
170 GET A$:IF A$="" THEN 170
180 END
200 LET A$="  ":PRINT CHR$(142)
210 POKE 53280,0:POKE 53281,0
220 FOR B=1 TO 2
230 PRINT"  ";
240 FOR S=1 TO 40:PRINT A$;:NEXT
250 PRINT"  ";
260 FOR S=1 TO 40:PRINT A$;:NEXT
270 PRINT"  ";
280 FOR S=1 TO 40:PRINT A$;:NEXT
290 NEXT B
300 RETURN
```

PROGRAM 23: MICRO-DICTIONARY

Program 23 turns your computer into a dictionary. This program lists only three words to show you how, but you can easily add more words.

The number of words to be stored is given at Line 20 as the variable NUM. This is also used in the arrays set up here.

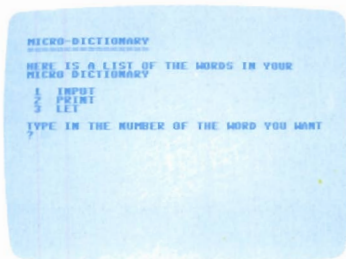
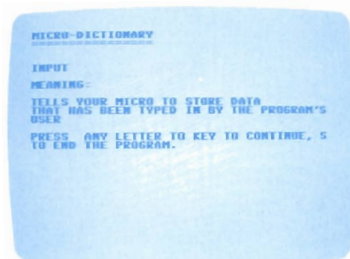
Lines 50 to 110 then read the information stored as DATA at Lines 250 to 330. The words, stored using the variable name WD\$, are printed out at this stage as a kind of contents page or MENU.

Line 120 invites you to choose the number of the word you want. Lines 140 to 190 then clear the screen and print out your word and its meaning. Line 200 allows you to carry on or finish the program.

The DATA is stored in three blocks. The first contains the word itself. The second two contain the meaning. This is split into two sections, because your computer won't accept DATA lines beyond a certain length.

Dividing up your DATA lines like this also makes your screen displays easier to plan. When planning screen layouts, you should always try to avoid having words chopped in half at the end of a line.

Look at Line 70, which shows one way of doing this. The words MICRO DICTIONARY are placed so that they begin directly underneath the word HERE. As HERE is at the beginning of the line, any words beginning under it will also start there. But if the words underneath overlap this point, those words will be split between two lines.



Type in:

```
10 REM * MICRO-DICTIONARY *
20 NUM=3: DIM WD$(NUM): DIM MG$(NUM): DIM
NMG$(NUM)
30 PRINT "G": POKE 53280, 12: POKE
53281, 12
40 PRINT CHR$(144)
50 PRINT "MICRO-DICTIONARY"
60 PRINT "=====": PRINT
70 PRINT "HERE IS A LIST OF THE WORDS IN
YOUR      MICRO-DICTIONARY"
80 PRINT: FOR Z=1 TO NUMBER
90 READ WD$(Z), MG$(Z), NMG$(Z)
100 PRINT Z " " WD$(Z)
110 NEXT Z
120 PRINT: PRINT "TYPE IN THE NUMBER OF
THE WORD  YOU WANT"
130 INPUT X
140 PRINT "G"
150 PRINT "MICRO-DICTIONARY"
160 PRINT "===== "
170 PRINT: PRINT WD$(X): PRINT
180 PRINT "MEANING: ": PRINT
190 PRINT MG$(X): PRINT NMG$(X)
200 PRINT "PRESS ANY LETTER KEY TO
CONTINUE,  PRESS S TO END"
210 GET A$: IF A$="" THEN 210
220 IF A$="S" THEN END
230 RESTORE: GOTO 30
240 REM * DICTIONARY DATA *
250 DATA "INPUT"
260 DATA "TELLS YOUR MICRO TO STORE
DATA"
270 DATA "THAT HAS BEEN TYPED IN BY THE
PROGRAM'S USER."
280 DATA "PRINT"
290 DATA "TELLS YOUR COMPUTER TO
DISPLAY"
300 DATA "INFORMATION ON THE VDU SCREEN"
310 DATA "LET"
320 DATA "TELLS YOUR MICRO TO STORE AN
ITEM OF"
330 DATA "DATA AS A VARIABLE"
```

PROGRAM 24:WORDSEARCH

The last program in this book is a computerised version of a 'wordsearch' puzzle.

After the program's introduction you will see a square made up of letters. Hidden in this square are sixteen words used in computing. These can be read horizontally, vertically and diagonally and can go up or down in all these directions.

If the person using the program thinks they can see a word, they type it in. The computer will tell them if theirs is a correct answer or not. To stop the program, type in the word STOP as an answer.

The program stores its data in two arrays - L\$, which stores each of the ten lines in the puzzle, and C\$, which stores the hidden words.

Line 30 sets up the arrays. Lines 40 to 120 print the program's introduction. Lines 130 to 170 print the puzzle.

Line 180 asks you to type in your word. Lines 190 to 220 then compare this with all the words stored in the array C\$. If the word typed in is the same as any of these words then the program moves to Line 270. This tells you that the answer is correct. After a short pause, the DATA is restored (reset), and the program returns to Line 130, so that you can have another go.

If the word typed in is not one of those stored in the array C\$, the program moves to Line 230. This first checks to see if the word STOP has been typed in. If it has, the program ends. If not, the program tells you that a wrong answer has been given, the DATA is reset and the program returns to Line 130.

This program could easily be altered. Decide on your theme, design your puzzle and type in the necessary data at Lines 300 to 390. Then make sure that the number of words hidden matches the size of the array C\$ and the loop set up at Line 190, and away you go!

P A E S R T L E M O
R D R A U C B I A G
I E T U L P M D S E
N U M E M O R Y O T
T C I N A K L O A B
Y U H E L E T H D M
P S J I P U K N A W
C L O O P E R N T D
F R E N U M L O A D
P S I T N E X T F L
TYPE IN YOUR WORD PLEASE

```

10 REM * WORDSEARCH *
20 PRINT "█"
30 DIM L$(10):DIM C$(16)
40 POKE 53280,13:POKE 53281,13
50 PRINT CHR$(28)
60 PRINT "WORDSEARCH"
70 PRINT "======"
80 PRINT:PRINT:PRINT "COMPUTERS":PRINT
90 PRINT "THERE ARE 16 COMPUTER WORDS
HIDDEN HERE"
100 PRINT "SEE HOW MANY YOU CAN FIND"
110 PRINT "PRESS ANY LETTER KEY TO
CONTINUE"
120 GET A$:IF A$="" THEN 120
130 POKE 53280,2:PRINT "█"
140 FOR N=1 TO 10
150 READ L$(N)
160 PRINT L$(N):PRINT
170 NEXT
180 INPUT "TYPE IN YOUR WORD PLEASE";WD$
190 FOR N=1 TO 16
200 READ C$(N)
210 IF C$(N)=WD$ THEN GOTO 270
220 NEXT N
230 IF WD$="STOP" THEN PRINT "GAME
OVER":END
240 PRINT "SORRY THAT WORD IS NOT HERE,
TRY AGAIN"
250 FOR P=1 TO 1000:NEXT P
260 RESTORE:GOTO 130
270 PRINT "WELL DONE, THAT IS CORRECT"
280 FOR P=1 TO 1000:NEXT P
290 RESTORE:GOTO 130
300 DATA "P A E S R T L E N O"
310 DATA "R D R A U C B I A G"
320 DATA "I E T V L P N D S E"
330 DATA "N U M E M O R Y O T"
340 DATA "T C I N A K L O A B"
350 DATA "Y U H E L E T H D M"
360 DATA "P S J I P U K N A W"
370 DATA "C L O O P E R N T D"
380 DATA "F R E N U M L O A D"
390 DATA "P S I T N E X T F L"
400 DATA "GET","LIST","PRINT","REM"
410 DATA "CHIP","MEMORY","POKE","LOOP"
420 DATA "INPUT","LOAD","NOT","DATA"
430 DATA "SAVE","NEXT","LEN","LET"

```




The programs you write can be 'saved', that is, recorded, on cassette tape. These taped programs can be used over and over again.

You cannot record programs from your Commodore 64 using an ordinary cassette recorder. The Commodore 64 uses its own special data cassette recorder. You will need to buy one of these if you want to load or save programs.

Connect the cassette unit to the computer. It plugs into a special socket at the back. Your cassette recorder does not need a separate power lead.

Put a blank tape in and make sure that it is wound past the start of the tape. Press the counter button so that it reads 000.

Before you start, make sure that you understand how to save and load programs.

Type in a short program to start with, then type in:

```
SAVE "
```

You now have to give your program a name, preferably one that describes it. For example, a program could be called "LAGOON". The quotation marks must be used.

On your screen you should now see this:

```
SAVE "LAGOON"
```

Press RETURN and you will see this message:

```
PRESS RECORD & PLAY ON TAPE
```

Now press the RECORD and PLAY buttons on your cassette unit. The screen will go blank for a short time. Next you will see the message:

```
SAVING LAGOON  
READY
```

This means that your program has been saved. You will see that the tape has automatically stopped running. Press the STOP button on your cassette unit.

LOADING

Now that you have saved your program, you can 'load' it back into your computer. First type in the command NEW, then press RETURN. This makes sure that your Commodore 64's memory is clear.

Rewind the tape back to the beginning and type in the message:

```
LOAD "LAGOON"
```


Press the RETURN key. You will now see this message:

PRESS PLAY ON TAPE

Press the PLAY button on your cassette recorder. The screen will go blank for a while and then you will see a message like this:

SEARCHING FOR LAGOON
FOUND LAGOON

Now press the Commodore key. The screen will again go blank for a short time. You will then see a message like this:

LOADING
READY

Type in the word LIST and you should see your program listed.

NAMING YOUR PROGRAMS

Always try to give your programs a name that describes them. Make a note of the number on the tape counter before you save your programs. This can help save time finding programs if you are storing several programs on tape.

If you type in the command LOAD on its own and then press RETURN, your Commodore 64 will LOAD the first program it finds on the tape.

An easy way of loading programs is to press the RUN STOP and SHIFT keys at the same time. When the FOUND message appears you don't need to press the Commodore key. The program will load and run automatically.

SUMMARY

When printing or storing words, quotation marks MUST be used:

```
PRINT "HELLO"  
LET A$="HELLO"
```

The final set of quotation marks may sometimes be left out. Where words are stored as variables, the variable name MUST be followed by a \$ sign:

```
LET B$="FRANKIE"
```

Variable names may be longer than single letters:

```
LET NAME$="HELEN"
```

Longer variable names beginning with the same two letters should not be used in the same program. Under certain other circumstances longer variable names may not be accepted, for example if they resemble words in BASIC. Longer variable names may be used in loops and arrays:

```
FOR DAY=1 TO 7  
DIM DAY$(7)
```

Words or strings may be stored in DATA statements without quotation marks. They may, however, be needed at certain times, such as when using BASIC words or mixing upper and lower case letters in DATA statements.

Used on its own, the word PRINT leaves a blank line. Using a comma following a PRINT statement leaves a gap of eight characters.

Using a semi-colon prints words or strings directly next to each other.

INDEX

- Abbreviations 11, 37
- Arrays 44-46
- ASCII code 42, 43
- Bugs 7-9
- Capital letters 21
- CHR\$ 43
- CLR HOME 12, 15
- Colour 12, 15, 18
- Commodore key 12, 21
- Control codes 12, 13, 18
- CRSR keys 9, 10
- CTRL key 13
- DATA 36, 39, 55
- DIM 45
- Editing programs 9-11
- GET 49-51
- Graphics shapes 13
- IF . . . THEN 22
- INPUT 17, 30, 48
- INST DEL 9, 10
- LEN 32, 33, 52
- LET 37
- Line numbers 5
- Listing 6, 7, 17
- Loading 61, 62
- Loops 23, 50
- MID\$ 33, 34
- Multi-statement lines 42
- NEW 11
- POKE 15
- PRINT 16, 18
- Punctuation 7, 16, 26
- READ 36
- REM statement 19
- RESTORE 12, 17, 50
- RETURN 5, 13
- RUN 6
- RUN STOP 6, 17
- RVS ON/OFF 12, 13, 30
- Saving 60, 61
- Scrolling 6
- SHIFT LOCK 12
- Slices 34
- Sort 46, 47
- Sound 27
- Spacing 16
- Strings 21
- Subscript 45
- Syntax error 7
- TAB 52
- Variables 15, 23, 63

PROGRAMS

Programs are listed under their REM statements.

Programs without REM statements are listed under their titles.

- ASCII Coding 43
- Back to Front 35
- Bubble Sort 47
- Fact File 20
- Friendly Start 19
- Getting your Message
 - Across 16
- Letter Count 33
- Lines 25
- Micro-dictionary 56
- Password 22
- READ and DATA 37
- Secret Code 44
- Simple Title 52
- Speedo 51
- Spelling Test 38
- Storyteller 31
- Thank you Letter 29
- Title Page 54
- Use of MID\$ 34
- Using DIM 46
- Using GET 50
- Word Order 41
- Word Pattern 26
- Wordsearch 59

MICRO MATES

Have fun with your computer!

These programs – they start simple and become progressively more complex – will give you lots of fun and teach you how to experiment with and enjoy your micro. You can adapt the programs as you like, and by the end of the book you will be able to write your own programs.

Make friends with your Commodore 64!

TITLES INCLUDE:

**SIMPLE PICTURES AND ANIMATION
SIMPLE MUSIC AND SOUND EFFECTS
SIMPLE WORDS AND WORD GAMES
SIMPLE FACTS AND FIGURES**